Effects of Introducing Survival Behaviours into Automated Negotiators

Peter Henderson, Stephen Crouch, Robert John Walters, Qinglai Ni
Declarative Systems and Software Engineering
Electronics and Computer Science
University of Southampton, Southampton, UK
{ph, stc, rjw1, qn}@ecs.soton.ac.uk

Abstract

With the rise of distributed e-commerce in recent years, demand for automated negotiation has increased. In turn, this has facilitated a demand for ever more complex algorithms to conduct these negotiations. As the complexity of these algorithms increases, our ability to reason about and predict their behaviour in an ever larger and more diverse negotiation environment decreases. In addition, with the proliferation of internet-based negotiation, any algorithm also has to contend with potential reliability issues in the underlying message-passing infrastructure. These factors can create problems for building these algorithms, which need to incorporate methods for survival as well as negotiation.

This paper proposes a simple yet effective framework for integrating survivability into negotiators, so they are better able to withstand imperfections in their environment. Results of an experiment are provided which show how the stability of a negotiation community is affected by incorporating an example survival behaviour into negotiators operating in an environment developed to support this framework.

1. Introduction

The choice of algorithm used to carry out automated negotiation on behalf of a client is a significant problem in distributed e-commerce [3, 4, 6-8, 12]. However, predicting how well a given algorithm will perform in a given environment is difficult.

The ability of an algorithm to succeed in an automated negotiation environment is dependent on its ability to survive in that environment. Automated negotiators, on their own terms, must be able to make sense of and conduct negotiation on the web in which there are no guarantees of the reliability of the underlying message-passing infrastructure. As the web increases in size and interconnectivity, this will become an even greater problem. In some cases, offers sent may not be received at all, but equally problematic is that offers received are out of date. Suppose a negotiator receives an offer that has spent an inordinate amount of time in transit. Despite replying promptly in sending an accept to this offer, the negotiator finds their acceptance rejected because the offer's sender has already sold their last stock to someone else. Agreement is not reached because of inconsistent views of negotiation, caused by inconsistent information. The ability to tolerate this information inconsistency, and being able to minimise its negative effects by taking corrective or compensating action, may reward the negotiator with greater success.

To compound these issues, a negotiator cannot be certain whether their experience of such problems with another negotiator is because of natural occurrence, or faulty or even malicious behaviour. It is also possible that the two negotiators are simply unable to reach agreement because they exhibit mutually incompatible negotiation strategies. To succeed, automated negotiators must be able to survive and progress despite such eventualities, without knowing the intent of other negotiators.

It is not uncommon for communities of automated negotiators to establish stable norms of behaviour. Over time, despite negotiators' different behavioural characteristics, initially erratic patterns of negotiation can eventually subside to more predictable patterns of co-operation. However, making successful predictions about how and in what form such stability will emerge can prove difficult [1-2]. Even more difficult are attempts to predict how the community will react when potentially disruptive elements are introduced into the environment.

Previously we have examined architectures for e-commerce systems [11-13], to investigate how federations of applications co-operate. We have also investigated the use of a fixed-length tournament-based approach to judge the fitness of negotiation algorithms against each other [10]. In this paper, we extend this approach to encompass the concept of a continuously operational environment, where negotiators may join and leave this community at any time. In our implementation of such an environment we are able to develop new algorithms using the

framework described in this paper, then introduce, observe and evaluate them as they participate in negotiations with others. We are also able to introduce uncertainty into the message-passing infrastructure, and observe how this affects the participants. Of particular interest is how these changes affect the stability of negotiation communities.

2. Reactive and proactive negotiation

In essence, the negotiation process consists of a number of offers being exchanged between two participants until agreement is reached. Essentially, notwithstanding the initial offer from either of the participants, this process is a reactive cycle: wait until an offer is received, evaluate the new offer, either reply with an acceptance, a counter-offer, or quit negotiations. This process is depicted in Figure 1. The dashed box at the top represents an initial action conducted by only one of the participants. This is the only proactive task in the process.

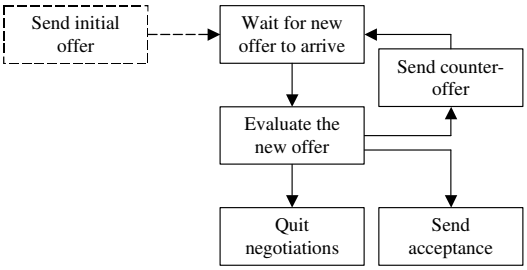

Figure 1. Reactive negotiation process

It is natural to assume that the structure of algorithms should follow this same rigid process. This idea is also easily extended to allow multiple negotiations with multiple participants.

In practice, adopting a purely reactive approach to the negotiation process is simply not sufficient. Developing negotiators in such a way does provide clarity of process, and simplicity of implementation. However, the success of the negotiator becomes ultimately dependent on the success of the negotiation process, which is itself dependent on the reliability of the operating environment. Notwithstanding 'bad' behaviour exhibited by negotiators, when this environment becomes unstable, the negotiation process is liable to collapse.

What is required is a more abstract, proactive approach to conducting negotiation. An approach that allows negotiators to reason about their circumstances at a higher level than the negotiation process alone, and adopt a more proactive view. Such a proactive approach should view negotiation as a fully manageable process: a means to achieve its objectives. In this way, a negotiator thusly accepts more responsibility for its survival and success, and reduces dependence on a potentially imperfect operating environment. Ideally, we would like the clarity offered by a reactive approach, coupled with the managerial power offered by the proactive approach.

In the context of this paper, we define success as a measurement of how well a negotiator *performs*, and we define survival as a measurement of whether a negotiator is able to *progress* in its environment. As will be observed later, recognising this separation of task success and environmental survival can enable more robust negotiators to be built.

3. The simulator

3.1. The car hire scenario

The scenario adopted for the negotiation simulation was car hire. If we consider a single participant in this environment, their objective is to secure a set number of 'hires' per day with respect to a given set of car specifications. For Buyers, we use a 'hire' unit to represent the number of cars they are in possession of for a given day. For Sellers, this unit represents the number of cars that are hired out for a given day.

The participant has their own set of example deals they would accept. Each entry in this set consists of two attribute name and value pairs for the following attributes:

- **Days** the length of time we wish to hire the car
- **Price** the price we would like to pay

A set of examples consists of a number of these pairings, each representing an acceptable outcome of negotiation. In practice, an instance of a Buyer participant within this framework uses their examples as deal targets for acquiring a number of hire cars, whilst an instance of a Seller participant views their examples as deal targets for hiring out available stock. A more realistic model of car hire would incorporate more attributes, (e.g. car size, car features, etc.), however the main objective of the experimentation is to observe overall communal behaviours, and so we have kept the model simple.

Specifying negotiation criteria as examples provides an abstract yet flexible method of stating a negotiator's desires, although the potential exists for ambiguity between these example criteria. There is not always a clear correlation between these examples, and the process of interpreting these examples in the context of the negotiation process is a task for the negotiator [14].

Table 1. A typical example set

Days	Price
9	250
6	100

Consider the example set in Table 1. If we assume they are a set of Buyer examples, we can easily determine

that they would be willing to pay 250 for 9 days of car hire, but also would pay significantly less on a cost per day basis for 6 days. In reality, such a discrepancy is often reasonable. It may be that the creator of this example set unavoidably requires a car for 9 days, so therefore ideally wants 9 days of car hire. If this were unavailable, the Buyer would be willing to accept 6 days of hire, but for a lot less per day to compensate for the extra effort of having to acquire 3 days of car hire after 6 days. If the Buyer receives an offer close to one of these examples, they would be inclined to accept it. If they have to make a counter-offer, the method they use takes into account their examples and offers received and attempts to stay close to them. If, for example, a Buyer negotiator who requires 2 cars per day were able to reach agreement for this quantity with a Seller for 9 days at 250, as in the set of examples, they would not need to negotiate again for another 9 days. If they only succeeded in obtaining 1 car for the same deal, they would need to attempt to find another deal somewhere else for the remaining car. The possibility exists that they will only be able to achieve their hires per day objective in part, or perhaps not at all.

3.2. The negotiation environment

An environment was developed which enables automated Buyers and Sellers to participate in the described scenario. This environment is similar in concept to a market run over an indefinite number of days; where Buyers and Sellers enter and leave at will on a daily basis. There are no restrictions on how many days they are able to participate, or how and with whom they conduct negotiations, although Buyers only negotiate with Sellers, and vice versa. Since there is no fixed duration to the simulation, negotiators cannot take advantage of other negotiators by exploiting the length of the simulation [5].

Participants are able to negotiate with anyone at any time. Their algorithm determines the manner in which they conduct negotiations with others to achieve their objectives. This allows us to construct and observe behaviour-rich simulations.

The environment consists of two components:

- **Supervisor** initiates, maintains and controls the environment, including the negotiators. Also maintains measures of negotiator performance.
- **Negotiator** given a set of negotiation parameters (including a set of examples and a target for the number of cars to possess/hire out each day), and is responsible for conducting negotiation.

To initiate a new environment, the Supervisor is launched, which then enables negotiators to be configured and instantiated, so they may participate in the simulation. Performance is measured by two factors: average hires per day, and average money spent/accrued per day. Each average calculation is based over the over last six days. This effectively gives us a running indicator of success (money per day) and survival (hires per day) as the simulation progresses. In short, if a negotiator is managing a high number of hires per day, it is surviving. If it manages a high amount of money for a seller - or a low amount for a buyer - per day, it is succeeding. Success and survivability are not only dependent on the reliability of communications, but on the structure of the community itself. If there is a shortage of car hire for sale, Buyers will do badly. If there is a shortage of demand, Sellers will do badly. In addition, in most cases the further apart the Buyer and Seller example sets are, the lower the likelihood of many deals being reached.

Figure 2 details the operation and message flow within the negotiation environment.

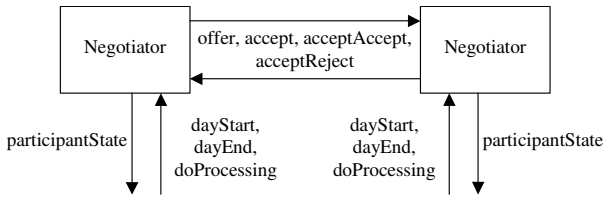

Figure 2. Operation and message flow within the negotiation environment

A message (e.g. offer) sent from one negotiator to another is stored in the receiving negotiators first-in-first-out 'message inbox', and it is each negotiator's responsibility to service their inbox and process its contents. Within this negotiation model, either negotiator (Buyer or Seller) may send an initial offer to the other. An arbitrary number of counter-offers are then subsequently made until one sends a request to accept their partner's last sent offer (*accept*). With each message type, a quantity is attached. For a Buyer, this represents the quantity of cars they want for that deal. For a Seller, this represents the quantity of cars they wish to hire out.

In such an asynchronous system, offers may become out of date; the negotiator may no longer be able to supply the quantity requested. Therefore, on receipt of an *accept*, an acknowledgement is required. This acknowledgement is either an *acceptAccept*, which is confirmation that the deal is accepted, or *acceptReject*, which is rejection. During this handshake, the quantity of resources stated in an *accept* are locked until either confirmation is received. This ensures that only one party accepts these resources.

How the Supervisor and Negotiator fit into this framework is examined in more detail in the following sections. Each specification follows an event-driven paradigm, specified using an abstract pseudocode.

3.2.1. Supervisor. The Supervisor coordinates the environment according to the following behaviour:

```
on startSimulation {
   do forever {
      for each participant in Participants {
         participant.dayStart()
      }
      for hours = 1 to 24 {
         randomize order of Participants
         for each participant in Participants {
            participant.doProcessing()
         }
      }
      for each participant in Participants {
         participant.dayEnd()
      }
   }
}
```

Here we can observe how the Supervisor manages the negotiators. Essentially, the simulation runs forever, and for each hour of each day, all the negotiators are requested to do a single 'chunk' of processing (*doProcessing()*). *dayStart()* and *dayEnd()* are called on each negotiator at the beginning and end of each day for performance measurement and maintenance purposes.

The *randomize order of Participants* statement introduces an element of fairness into the simulation. Without this statement, each *participant* in the list of *Participants* would always be called in the same order; the simulation would perhaps favour the first negotiator, since he has a greater chance of securing the first deal.

3.2.2. Negotiator. A Negotiator developed in the framework requires two distinct areas of its behaviour to be specified: reactive and proactive. The reactive aspect of its behaviour handles the negotiation process:

```
on receive offer from participant {
   // What to do when an offer is received
}
on receive accept from participant {
   // Agree to an acceptance proposal?
   // (i.e. the other participant wishes to
   // accept your last offer)
   // Returns either True or False
}
on receive acceptAccept from participant {
   // What to do when an acceptance to a
   // previously sent accept is received. Here,
   // negotiation is positively concluded with
   // 'participant'
}
on receive acceptReject from participant {
   // What to do when a reject to a previously
   // sent accept is received. Here, negotiation
   // is negatively concluded with 'participant'
}
```

The proactive aspect of the negotiator's behaviour handles survival issues, assessing its situation and instigating corrective actions based on negative aspects of this assessment:

```
on algorithmProcess {
   // proactive behaviour is specified here
}
```

This conditionally proactive behaviour is specified as rules, in the form of condition -> action pairs. As will be demonstrated later with an example, we are able to specify survival behaviours within this function.

When the Supervisor invokes a negotiator's *doProcessing()* instruction, the framework performs two actions transparent to the algorithm's creator:

- **Reactive - Service message box** new messages are read and the negotiator's reactive functions are invoked depending on the message type.
- **Proactive - Invoke algorithm's proactive function (*algorithmProcess*)** this allows the negotiator the opportunity to evaluate their situation, and possibly take proactive action.

Thus, in this framework, there is a distinct separation between the reactive and proactive parts of the negotiation process. Suppose a negotiator initiates negotiation with another negotiator by sending them an initial offer. From this point, the negotiation process is dealt with by the reactive functions; multiple negotiations with multiple partners are handled automatically. However, the algorithm is still able to take proactive action if required, enabling the negotiator to monitor and *manage* these negotiation processes at a more abstract level.

4. The experiment

4.1. Overview

This paper will detail one experiment, which consists of four simulations. Other experiments were conducted which used different algorithms and different example sets. The results presented here are representative of these other experiments.

The first simulation establishes how a community of negotiators develops in a stable message-passing environment. The negotiators are naïve in that they adopt a purely reactive approach to negotiation.

In the second simulation, it is established how the same community of negotiators develops with an element of uncertainty introduced into the message-passing environment. In this simulation, there is a 10% chance that a sent offer will not reach its destination.

For the third simulation, it will be shown how each algorithm can be adapted to incorporate an example proactive survival behaviour, and the simulation executed again with reliable communications. This provides us with an opportunity to observe how these adapted algorithms behave in a reliable environment.

The fourth simulation illustrates how these adapted algorithms behave in the unreliable communications environment.

After each day, the performance of each negotiator is evaluated, with respect to average hires per day and average money accrued (Seller) or spent (Buyer) per day.

Thus, we have measures of survival and success respectively. This paper focuses on the results for average hires per day, since we wish to examine the survivability of the community. By examining how the number of hires per day for each participant changes over time, we are able to reason about the survivability of the negotiation processes conducted by the negotiators, and ultimately, the survivability of the negotiation community.

4.2. The algorithms

In each simulation, instances of two algorithms form the negotiation community. These algorithms were not designed to be realistic negotiators. Rather, their strategies are designed to be sufficiently simple that we are able to reason about their behaviour in a complex, evolving community.

- **Stubborn** this mimics 'stubborn', anti-concessionary behaviour by sending only its examples as offers to a negotiation partner. It does not attempt to reason about the offers received. Initially, it sends its first example, then its second, etc. When it has sent all its examples, it starts again. After 30 rounds of negotiation, it simply accepts the last received offer from its partner.
- **Experimental** this is a far more reactive algorithm than stubborn. Its first example forms its initial offer. When a new offer is received, it attempts to find an example that matches the number of days in the received offer. If found, it sends a sequence of offers for this example, each a little more concessionary on price. If not found, it does the same but with the first example. When it reaches a $20 concession on price for the selected example, it attempts to move negotiation to its next example by specifying it as the next offer.

Initially, a simulation begins with 2 'Stubborn' buyers and 2 'Stubborn' sellers conducting negotiations. After 20 days of negotiation, an 'Experimental' buyer and 'Experimental' seller are introduced to observe how this affects the community.

Each negotiator is responsible for a number of negotiation processes; a maximum of one per partner at any given time. As a result, the possibility exists that negotiations with one party will be abandoned in favour of accepting another deal from another party. However, this does not prevent the original two negotiators from resuming negotiations from where they were abandoned at a later date, if both are prepared to do so. The algorithms present in this experiment do exhibit this forgiving behaviour, not forcing negotiations with the one who abandoned the negotiation to begin at the start of their behavioural process. As will be observed in the next chapter, this leads to some interesting behaviour.

4.3. The example sets

Each Buyer is given the same set of Buyer examples, and each Seller given the same set of Seller examples. The Buyer example set is given in Table 2 and the Seller example set is given in Table 3.

Table 2. Buyer example set

Days	Price
2	160
4	240

Table 3. Seller example set

Days	Price
1	100
2	180
3	200

The objective of each Stubborn Buyer and Seller is to make one deal per day, whilst the objective of the Experimental Buyer and Sellers is to make four hires per day. However, this is very difficult for the Stubborn negotiators to achieve, due to the length of their negotiation process. It is not impossible, however, as will be observed in the results in the next chapter.

5. Results

Figure 3 shows the results of simulation 1, representing average hires per day for each negotiator over a 40 day period. The four lines clustered at the bottom represent the four Stubborn negotiators, whilst the two at the top represent the two Experimental negotiators.

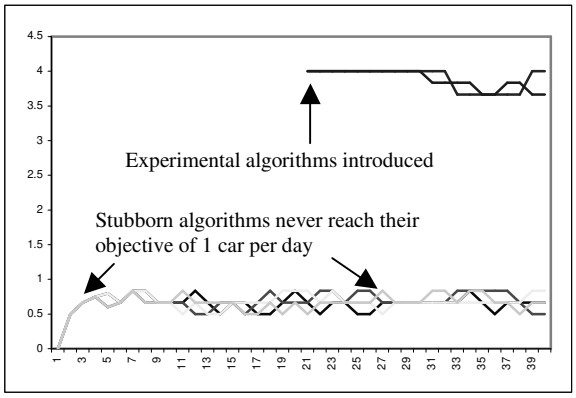

Figure 3. Simulation 1 - average hires per day

Since Stubborn always takes 30 rounds (hours) to reach a deal, we observe that there are no deals struck on the first day. After about 4 days, in which all negotiators perform equally well, these negotiators begin to exhibit a certain communal behaviour. Note that no Stubborn reaches their maximum hires per day. This is due to the inflexible, laborious nature of the Stubborn algorithm.

One might expect a strict pattern to be observed. However, there are two reasons why this is not the case.

Firstly, the environment has been designed with fairness in mind: as mentioned in section 3.2.1 the order in which negotiators perform processing is random. Secondly, participants are able to abandon negotiations with a particular party, and resume them at a later date (see section 4.1). Together, these factors decrease the likelihood that such behavioural harmonics will occur.

Also interesting are the results of the two Experimental algorithms. Following their introduction after 20 days, they remain very stable in their behaviour, in fact achieving their maximum required hires per day for nearly 10 days. After this however, a significant slump occurs. This is because initially, they deal only with each other; their negotiation process is far more efficient than Stubborn's, and they reach agreement quickly. Following this, however, one of them chances to strike a deal with one of the Stubborn negotiators, and this causes them to be affected by their erratic negotiation strategy. The reason that one of the Experimental negotiators makes this choice, despite it resulting in decreased hires per day, is that they are not able to anticipate that this will have a negative impact later. They make a choice that appears optimal at the time, but have no way of knowing that it represents a poor global choice.

Figure 4 illustrates the effects of introducing a 10% chance of an offer being lost. The results are as expected. Clearly, the Stubborn negotiators are unable to reach agreement at all. After the two Experimentals are introduced, they do slightly better due to their more efficient negotiation process. They reach agreement quickly, but following this, they are unable to maintain their initial success. Their negotiation processes also become affected by their inability to reason about their failure and take corrective action.

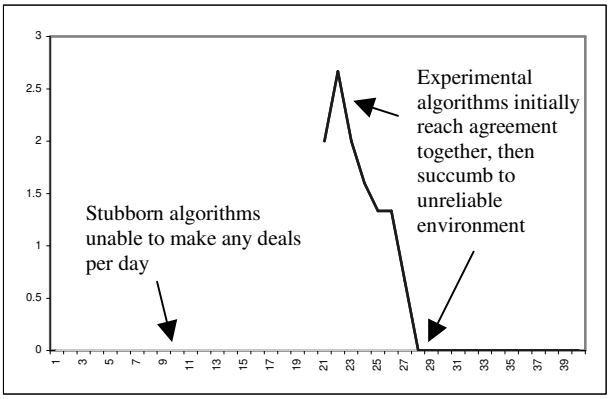

Figure 4. Simulation 2 - average hires per day

In order to conduct the third simulation, a survival behaviour had to be integrated into both the algorithms. The example survival behaviour chosen was a timeout. The behaviour was specified as follows:

```
on algorithmProcess {
   for each participant in Participants {
      if (have sent offer to Participant &
        have not had reply in least 3 hours) {
         send last sent offer to participant
      }
   }
}
```

Essentially, if we have not received a response to an offer sent in the last 3 hours, just resend the offer again. The results of simulation 3 are given in Figure 5.

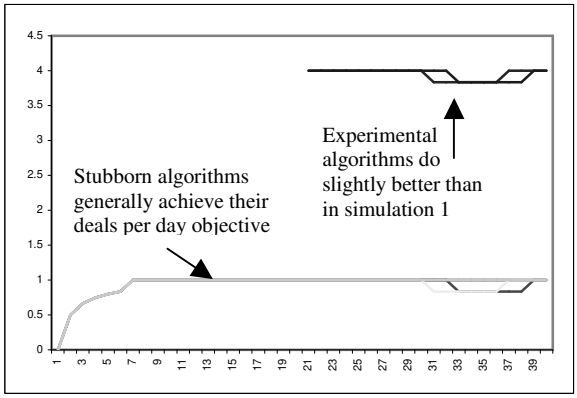

Figure 5. Simulation 3 - average hires per day

The results were not as expected. Introducing this survival behaviour into this system has maximised the efficiency of the community; they are achieving greater hires per day on average. Even more surprising is the stability that has clearly emerged among the Stubborn negotiators. These effects are being observed because the survival behaviour has affected the negotiation process of the algorithms. This behaviour essentially takes action in response to unresponsive negotiators. So, even under normal circumstances, it is still likely this behaviour will be instigated. Effectively, a negotiator now responds to abandonment from another negotiator by resending their last offer. This is discussed in more detail later.

As with the first simulation, both algorithm groups initially reach agreement only with those of their own type, but are eventually tempted by negotiations with the other group. Here we clearly observe that this behaviour affects the stability of the community as a whole.

Figure 6 displays the results of simulation 4. Surprisingly, introducing a 10% message loss to the more robust community of negotiators has not dramatically affected the survival of their community. In fact, despite an initial inability to attain the stability noticed in simulation 3, the community eventually achieves a similar degree of survival. More significantly, this community achieves a greater degree of survival than observed in the first simulation. The result that the experimental algorithms always achieve their objectives is anomalous in this case, and not representative.

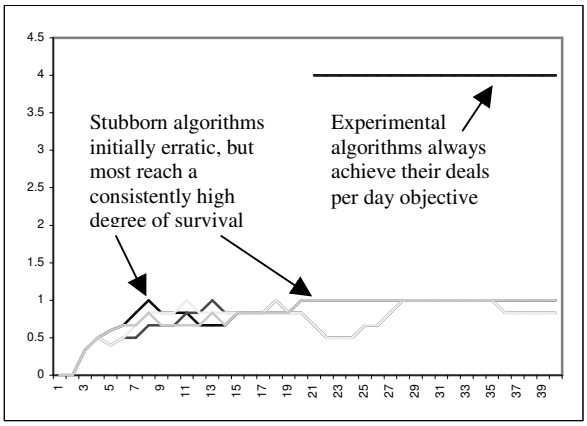

Figure 6. Simulation 4 - average hires per day

Why is it that the negotiators in *both* simulations 3 and 4 survive so much better? The reason is because we have introduced *chaos* and *opportunity* into the community. By introducing the survival behaviours into these algorithms, we have improved their ability to survive in an unreliable environment. But, we have also made their individual negotiation processes more agile; they no longer follow a rigid reactive path to a negotiation's conclusion. But, as a result, the negotiation process that occurs between two negotiators is less predictable. If we extend this perspective and observe the actions of the community as a whole, it has become more chaotic. However, this chaos has introduced a very welcome factor: opportunity. Negotiators are making more offers to their partners. This increases the chances that their quota is achieved.

6. Conclusion

Developing robust automated negotiators able to survive and succeed in the complex, evolving environment of the internet will become increasingly difficult. The ability to evaluate progress and proactively take corrective or compensatory action outside of the rigid negotiation process confers a greater degree of survival.

This paper has introduced a framework that enables the developer to specify reactive and proactive behaviour separately. Developers are able to specify conditional behaviour at a higher level than the negotiation process that is able to manage and optimise this process. By integrating an example survival behaviour into the algorithms we have demonstrated that not only has this improved their ability to take proactive action in cases of suspected message loss, but as a side-effect have made their negotiation processes more flexible. By responding to ineffectual negotiations at this abstract level with our example survival behaviour, we have created more opportunities for success. Negotiators are no longer strictly adhering to the rigidity of the reactive negotiation process; they proactively increase their potential to make more deals by maximising efficiency within this process.

With the need to increase the robustness of negotiators in an ever more complex environment, we need to be able to predict how this robustness will affect their behaviour with others. When implementing self-protection measures to ensure survival, we need to know that we are not hindering the negotiator's ability to succeed.

References

[1] R. Axelrod, *The Evolution of Co-operation*, Basic Books Inc., New York, 1984.

[2] R. Axelrod, *The Complexity of Cooperation*, Basic Books Inc., New York, 1997.

[3] B. Burg, "Agents in the World of Active Web Services", *Lecture Notes in Computer Science*, Vol. 2362, Springer, 2002, pp. 343-356.

[4] M. Bichler, A. Segev, J.L. Zhao, "Component-Based E-Commerce: Assessment of Current Practices and Future Directions", *ACM Sigmod Record: Special Section on Electronics Commerce*, Vol. 27, No. 4, 1998, pp. 7-14.

[5] K. Binmore, N. Vulkan,, "Applying Game Theory to Automated Negotiation", *Netonomics*, Jan. 1999. See http://www.worcester.ox.ac.uk/fellows/vulkan.

[6] L.F. Cranor, P. Resnick, "Protocols for Automated Negotiations with Buyer Anonymity and Seller Reputations", *Proceedings of the Telecommunications Policy Research Conference (TPRC 97)*, 1997. See http://www.si.umich.edu/~presnick.

[7] F. Farhoodi, P. Fingar, "Developing Enterprise Systems with Intelligent Agent Technology", *Distributed Object Computing*, Object Management Group, 1997.

[8] P. Fingar, H. Kumar, T. Sharma, *Enterprise E-Commerce*, 1st edn. Meghan-Kiffer Press, Tampa FL, 2000.

[9] D.B. Fogel, "Applying Fogel and Burgin's 'Competitive Goal-Seeking through Evolutionary Programming' to Coordination, Trust and Bargaining Games", *Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000)*, IEEE Press Piscataway NJ, 2000, pp. 1210-1216.

[10]. P. Henderson, R.J. Walters, S. Crouch, Q. Ni, "A Comparison of some Negotiation Algorithms using a Tournament-Based Approach", *Lecture Notes for Computer Science*, Vol. 2592, Springer-Verlag Berlin Heidelberg, 2003, pp. 137-150.

[11] P. Henderson, "Laws for Dynamic Systems", *Proceedings of the Fifth International Conference on Software Reuse (ICSR 98)*, IEEE Computer Society Press, 1998, pp. 330–336. See http://www.ecs.soton.ac.uk/~ph/papers.

[12] P. Henderson, R.J. Walters, "Behavioural Analysis of Component-Based Systems", *Information and Software Technology*, Vol. 43, No. 3, 2001, pp. 161–169.

[13] P. Henderson, "Asset Mapping - Developing Inter-enterprise Solutions from Legacy Components", In: *Systems Engineering for Business Process Change - New Directions*, Springer-Verlag UK, 2002, pp. 1-12.

[14] R. Sesseler, "Building Agents for Service Provisioning out of Components", *Proceedings of the Fifth International Conference on Autonomous Agents*, 2001.